ONLINE DEGREE-BOUNDED STEINER NETWORK DESIGN

Sina Dehghani
Saeed Seddighin
Ali Shafahi
Fall 2015
Online Steiner Forest Problem

- An initially given graph $G$.
- A sequence of demands $(s_i, t_i)$ arriving one-by-one.
- Buy new edges to connect demands.
There is a given bound \( b_v \) for every vertex \( v \).

degree violation := \( \frac{\text{deg}_H(v)}{b_v} \).

Find a Steiner forest \( H \) minimizing the degree violations.
## Previous Offline Work

- **Degree-bounded network design:**

<table>
<thead>
<tr>
<th>Problem</th>
<th>Paper</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Degree-bounded Spanning tree</td>
<td>FR '90</td>
<td>$O(\log n)$-approximation</td>
</tr>
<tr>
<td>Degree-bounded Steiner tree</td>
<td>AKR '91</td>
<td>$O(\log n)$-approximation</td>
</tr>
<tr>
<td>Degree-bounded Steiner forest</td>
<td>FR '94</td>
<td>maximum degree $\leq b^* + 1$</td>
</tr>
</tbody>
</table>
### Previous Offline Work

- **Edge-weighted degree-bounded variant:**

<table>
<thead>
<tr>
<th>Problem</th>
<th>Paper</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>EW DB Steiner forest</td>
<td>MRSRRH. ’98</td>
<td>(O(\log n), O(\log n))-approx.</td>
</tr>
<tr>
<td>EW DB Spanning tree</td>
<td>G ’06</td>
<td>min weight, max deg (\leq b^* + 2)</td>
</tr>
<tr>
<td>EW DB Spanning tree</td>
<td>LS ‘07</td>
<td>min weight, max deg (\leq b^* + 1)</td>
</tr>
</tbody>
</table>
Previous Online Work

- Online weighted Steiner network (no degree bound)

<table>
<thead>
<tr>
<th>Problem</th>
<th>Paper</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Online edge-weighted Steiner tree</td>
<td>IW '91</td>
<td>$O(\log n)$-competitive</td>
</tr>
<tr>
<td>Online edge-weighted Steiner forest</td>
<td>AAB '96</td>
<td>$O(\log n)$-competitive</td>
</tr>
</tbody>
</table>
## OUR CONTRIBUTION

- Online degree-bounded Steiner network:

<table>
<thead>
<tr>
<th>Problem</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Online degree-bounded Steiner forest</td>
<td>$O(\log n)$-competitive greedy algorithm</td>
</tr>
<tr>
<td>Online degree-bounded Steiner tree</td>
<td>$\Omega(\log n)$ lower bound</td>
</tr>
<tr>
<td>Online edge-weighted degree-bounded Steiner</td>
<td>$\Omega(n)$ lower bound</td>
</tr>
<tr>
<td>tree</td>
<td></td>
</tr>
<tr>
<td>Online degree-bounded group Steiner tree</td>
<td>$\Omega(n)$ lower bound for det. algorithms.</td>
</tr>
</tbody>
</table>

∀e ∈ E: x(e) = 1 if and only if e is selected.

S be the collection of separating sets of demands.

OMPC has an $O(\log^2 n)$-competitive fractional solution, but rounding that is hard!

\[
\begin{align*}
\min \alpha \\
\forall v \in V \sum_{e \in \delta(v)} x(e) &\leq \alpha \cdot b_v \\
\forall S \in S \sum_{e \in \delta(S)} x(e) &\geq 1
\end{align*}
\]

limits degree violations.

ensures connectivity.
REDUCTION TO UNIFORM DEGREE BOUNDS

- Replace $v$ with $v_1 \ldots v_{b_v}$.
- Connect each $v_i$ to all neighbors of $v$.
- Set all degree bounds to 1.
- Uniformly distribute edges of $\delta_H(v)$ among $v_i$'s.
- The degree violation remains almost the same.
GREEDY ALGORITHM
GREEDY ALGORITHM

Definitions:
- Let $H$ denote the online output of the previous step.
- For an $(s, t)$-path $P$ the extension part is $P^* = \{e | e \in P, e \notin H\}$.
- The load of $P^*$ is $l_H(P^*) = \max_{v \in P^*} \deg_H(v)$.

Algorithm:
1. Initiate $H = \phi$.
2. For every new demand $(s_i, t_i)$:
   1. Find the path $P_i$ with the minimum $l_H(P_i^*)$.
   2. $H = H \cup P_i^*$.
ANALYSIS

Let $\Gamma(r)$ be the set of vertices with $\deg_{H}(v) \geq r$.

Let $D(r)$ be demands for which $l_{H}(P_{i}^{*})$ is at least $r$.

Remark: $\Gamma(r)$ is a cut-set for $s_{i}$ and $t_{i}$ for every $i \in D(r)$.

Let $CC(r)$ denote the number of connected components of $G \setminus \Gamma(r)$ that have at least one endpoint of demand $i \in D(r)$.

Lemma: $\forall r$: $CC(r) \geq |D(r)| + 1$.

Remark: $\forall r$: $OPT \geq \frac{CC(r)}{|\Gamma(r)|}$.
ANALYSIS

- $\Gamma(r)$'s have a hierarchical order, i.e. $\Gamma(r + 1) \subseteq \Gamma(r)$.

- Every demand $i \in D(r)$ copies some vertices to upper level.

- Out of all copies, at most $2(\Gamma(r) - 1)$ are for internal edges.

**Lemma:** $\forall r: |D(r)| \geq \sum_{t=r+1}^{A} |\Gamma(t)| - 2(\Gamma(r) - 1)$. 
Lemma: For every sequence of integers $a_1 \geq a_2 \geq \cdots \geq a_\Delta > 0$

$$\max_i \left\{ \frac{\sum_{j=i}^{\Delta} a_j}{a_i} \right\} \geq \frac{\Delta}{2 \log a_1}.$$ 

- Partition to $\log a_1$ groups.
- One group has at least $\frac{\Delta}{\log a_1}$ numbers.
Putting all together:

\[ \forall r: OPT \geq \frac{CC(r)}{|\Gamma(r)|} \Rightarrow OPT \geq \max_r \frac{CC(r)}{|\Gamma(r)|}. \]

\[ CC(r) \geq |D(r)| + 1. \]

\[ D(r) \geq \sum_{t=r+1}^{\Delta} |\Gamma(t)| - 2(\Gamma(r) - 1). \]

Setting \( a_i = |\Gamma(i)| \) and using the lemma:

\[ OPT \geq \max_r \frac{\sum_{t=r}^{\Delta} |\Gamma(r)| - o(|\Gamma(r)|) + 1}{|\Gamma(r)|} \geq \frac{\Delta}{2 \log |\Gamma(1)|} - O(1) \in \Omega\left(\frac{\Delta}{\log n}\right) \]
**Theorem:** Every (randomized) algorithm for online degree-bounded Steiner tree is \( \Omega(\log n) \)-competitive.

\[
\begin{align*}
n &\in O(2^{(2^l)}) \\
b_v &= \begin{cases} 
  n & \text{if } v = \text{root} \\
  2 & \text{otherwise}
\end{cases}
\end{align*}
\]
**Theorem:** Let $OPT_b$ denote the minimum weight of a Steiner tree with maximum degree $b$. Then for every (randomized) algorithm $A$ for online edge-weighted degree-bounded Steiner tree either

- $E[\max \deg_A(v)] \geq \Omega(n) \cdot b$
- or
- $E[\text{weight}(A)] \geq \Omega(n) \cdot OPT_b$.

\[ n = 2k + 1 \]
\[ b = 3 \]
\[ \text{weight}(A) = n^{i+1} \]
\[ \deg_A(\text{root}) = i \]

\[ OPT_3 = \sum_{j=1}^{i} n^j \in O(n^i) \]
**Theorem:** Every deterministic algorithm $A$ for online degree-bounded group Steiner tree is $\Omega(n)$-competitive.

All degree bounds are 1.

$\deg_A(root) = n - 1$. 
OPEN PROBLEMS

➢ The main open problem:
  ▪ Online edge-weighted degree-bounded Steiner forest,
    when the weights are polynomial to $n$.

➢ Other degree-bounded variants (with or without weights):
  ▪ Online group Steiner tree.
  ▪ Online survivable network design.
Thank you